Toteutusdokumentti

Toteutuksessa tehtiin kolme erilaista selection algoritmia, kolme replace algoritmia sekä yksi yhdistetty algoritmi joka hoitaa molemmat tehtävät (FloodFill + MixedRepeat).

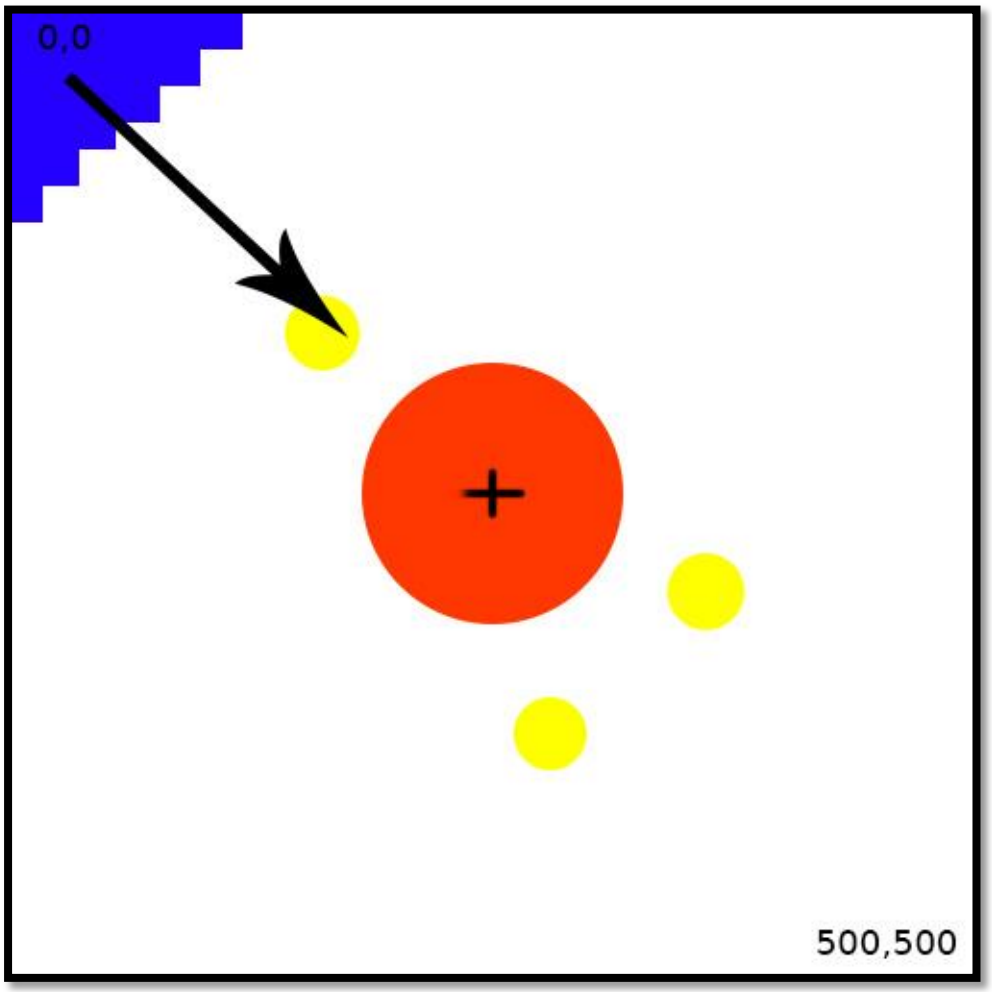
Aika- ja tilavaativuuksissa n = kuvan pikseleiden määrä.

Selection algoritmit:

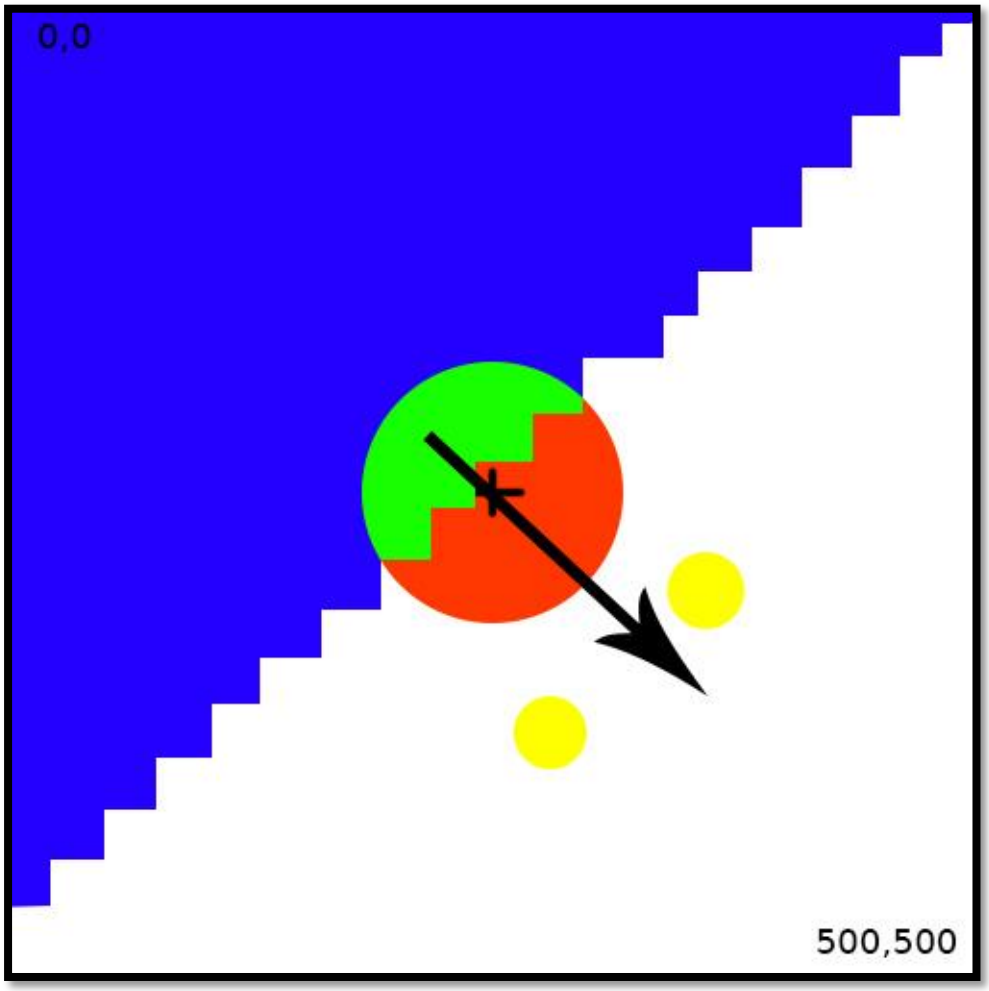
DummyLoop

Aika: O(n), Tila: O(n)

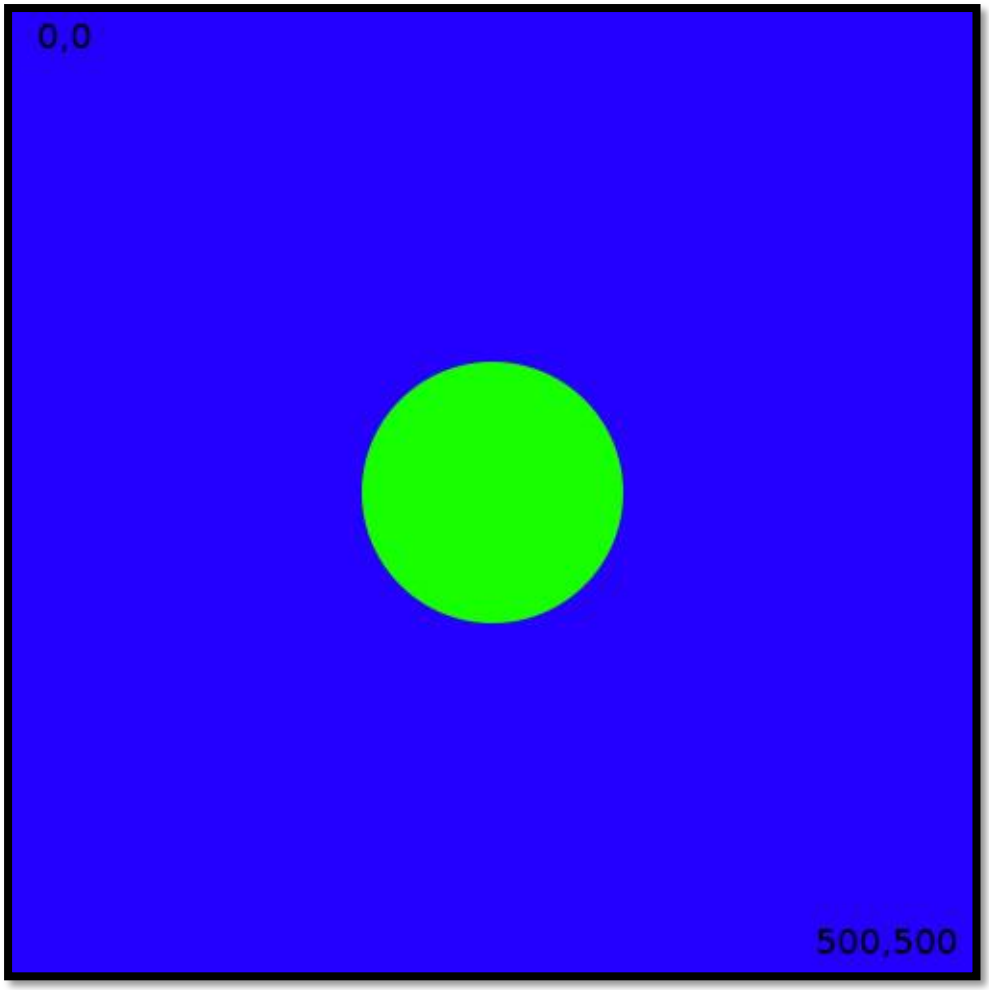
Algoritmi käy kaikki kuvan pikselit lävitse. Risti punaisen ympyrän keskellä tarkoittaa klikkaus kohtaa.



Jos pikselin väri eroaa klikkaus kohdan väristä riittävän vähän, merkitään pikseli valituksi (vihreä väri).



Kun kaikki pikselit on käyty läpi, palautetaan vihreä alue.



Hyvät ja huonot puolet:

+ Suhteellisen nopea (O(n))

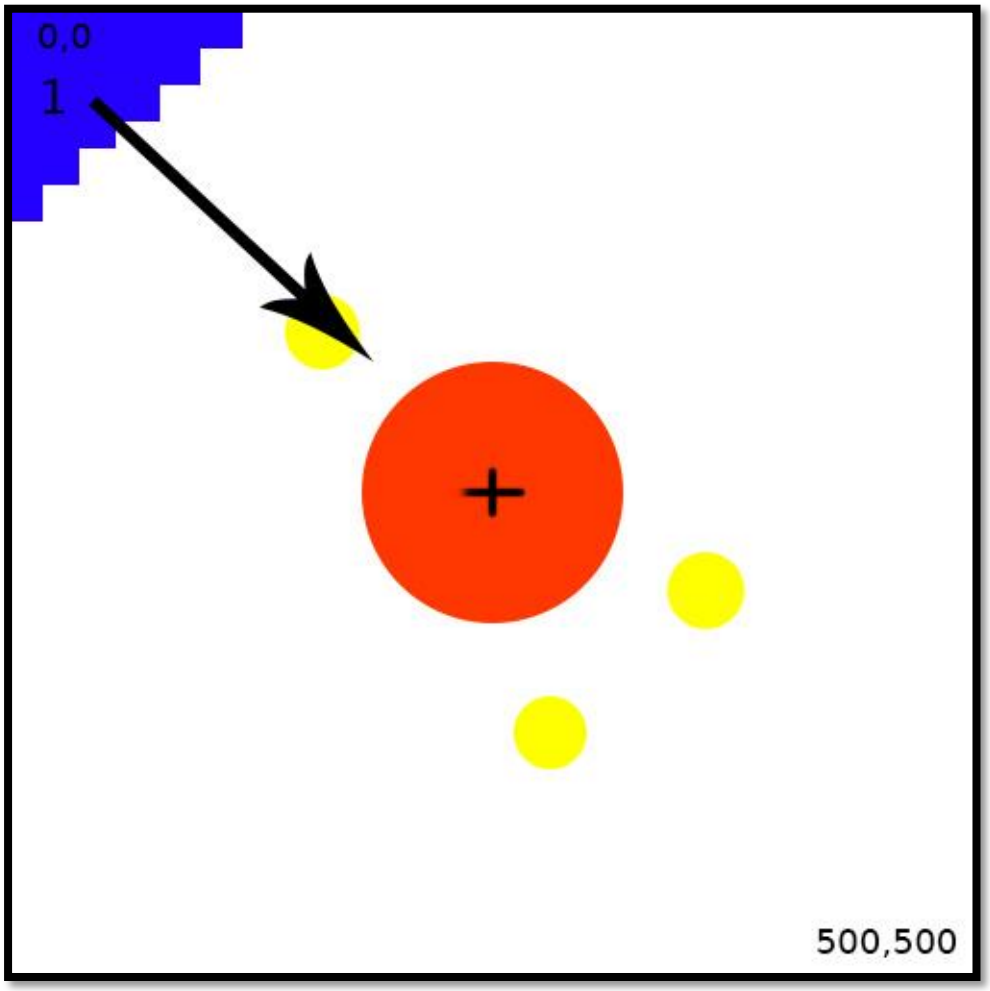
- Kaikki pikselit pitää käydä läpi

- Algoritmi valitsee myös saman väriset alueet jotka eivät ole lähelläkään klikkaus kohtaa

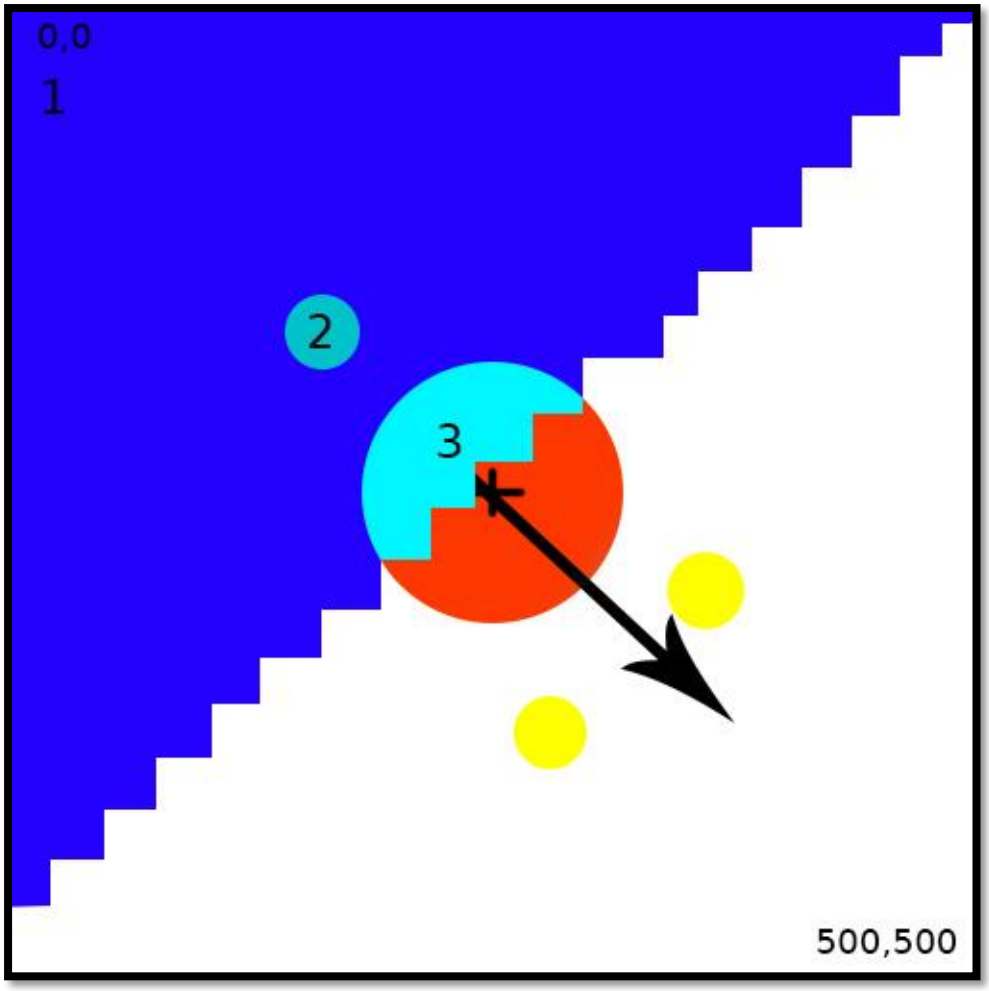
One-pass Connected-component labeling

Aika: O(n^2), Tila: O(n), hyvä lopputulos vaatii > 2n läpikäyntiä.

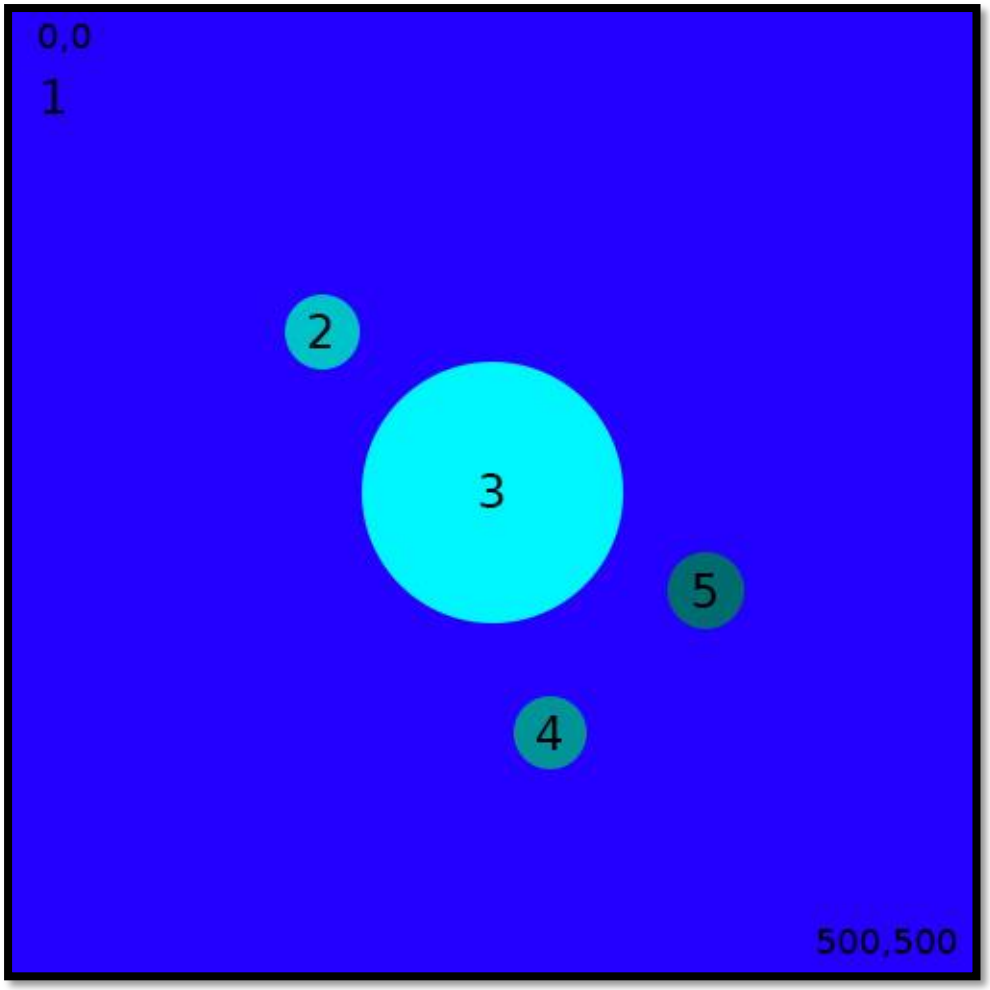
Algoritmi käy kaikki pikselit lävitse. Jokainen uusi väri merkitään omaksi alueekseen.



Algoritmi jatkaa pikseleiden läpikäyntiä ja merkitsee jokaisen värin omaksi alueekseen.



Lopputuloksena saadaan taulukko jossa jokainen alue on merkitty omalla numerolla.



Hyvät ja huonot puolet:

+ Jos kyseessä on staattinen kuva (kuten tässä demossa) ja aluetta tulisi vaihtaa useasti, onnistuu se nopeasti yhden läpikäynnin jälkeen

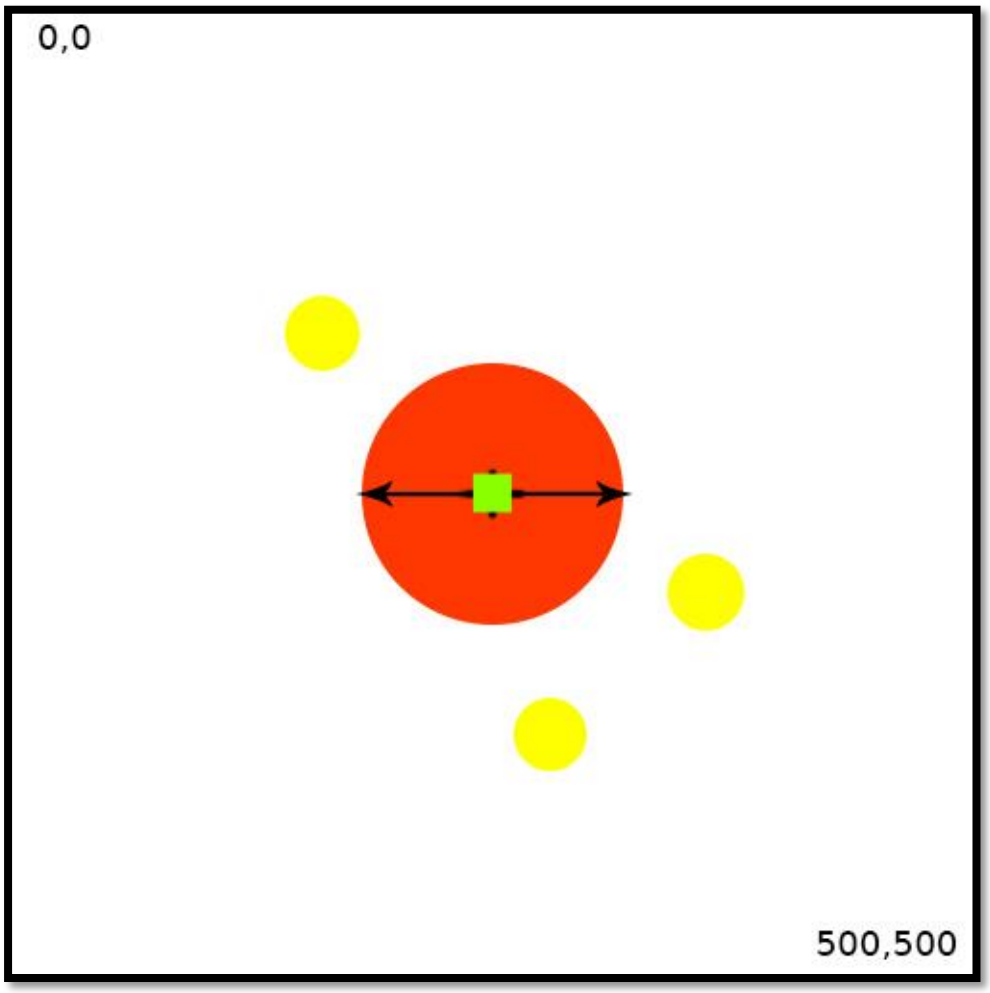
- Algoritmi tulisi suorittaa useita kertoja ennen kuin alueet saadaan kunnolla kartoitettua (huomattavasti hitaampi kuin muut)

FloodFill

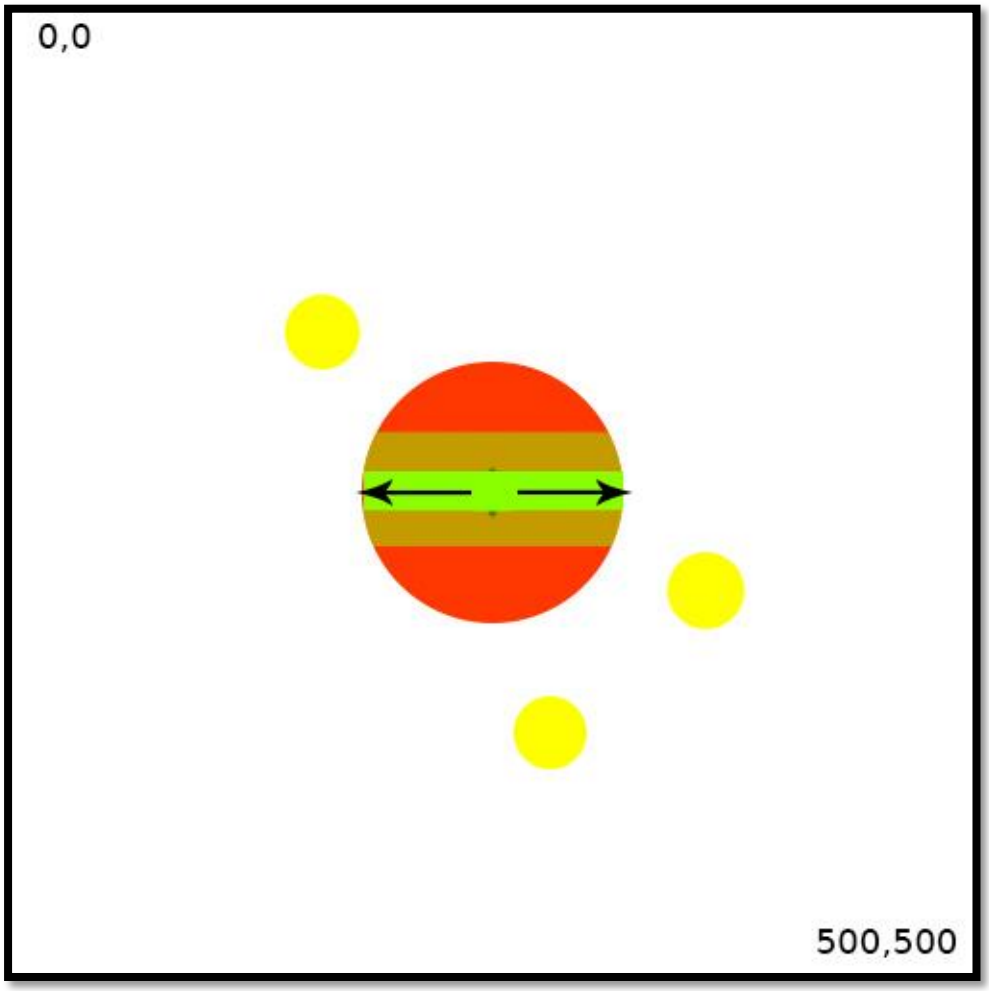
Aika: O(n), Tila: O(n), mutta yleensä aikaa kuluu huomattavasti vähemmän

Tässä algoritmissa pikseleiden läpikäynti aloitetaan klikkaus kohdasta.

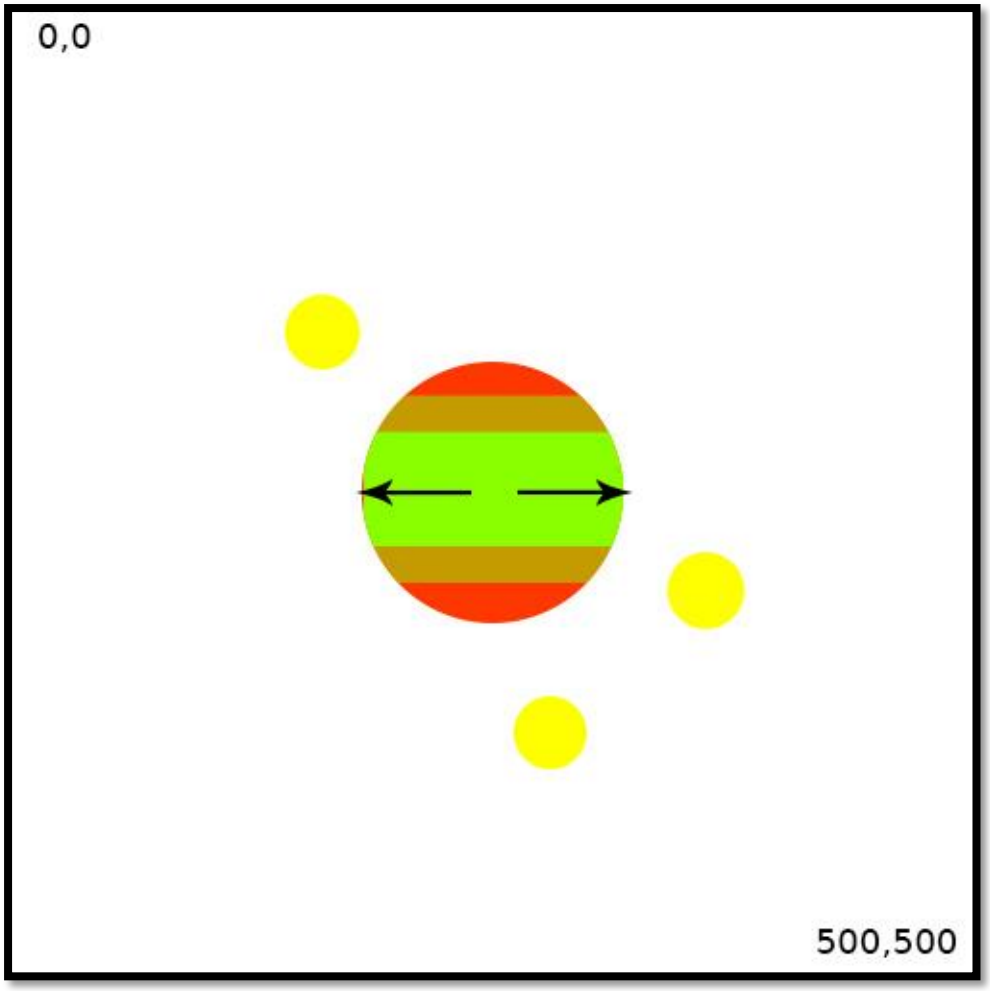
Algoritmi etsii aluksi valitun alueen vasemman ja oikean reunan.



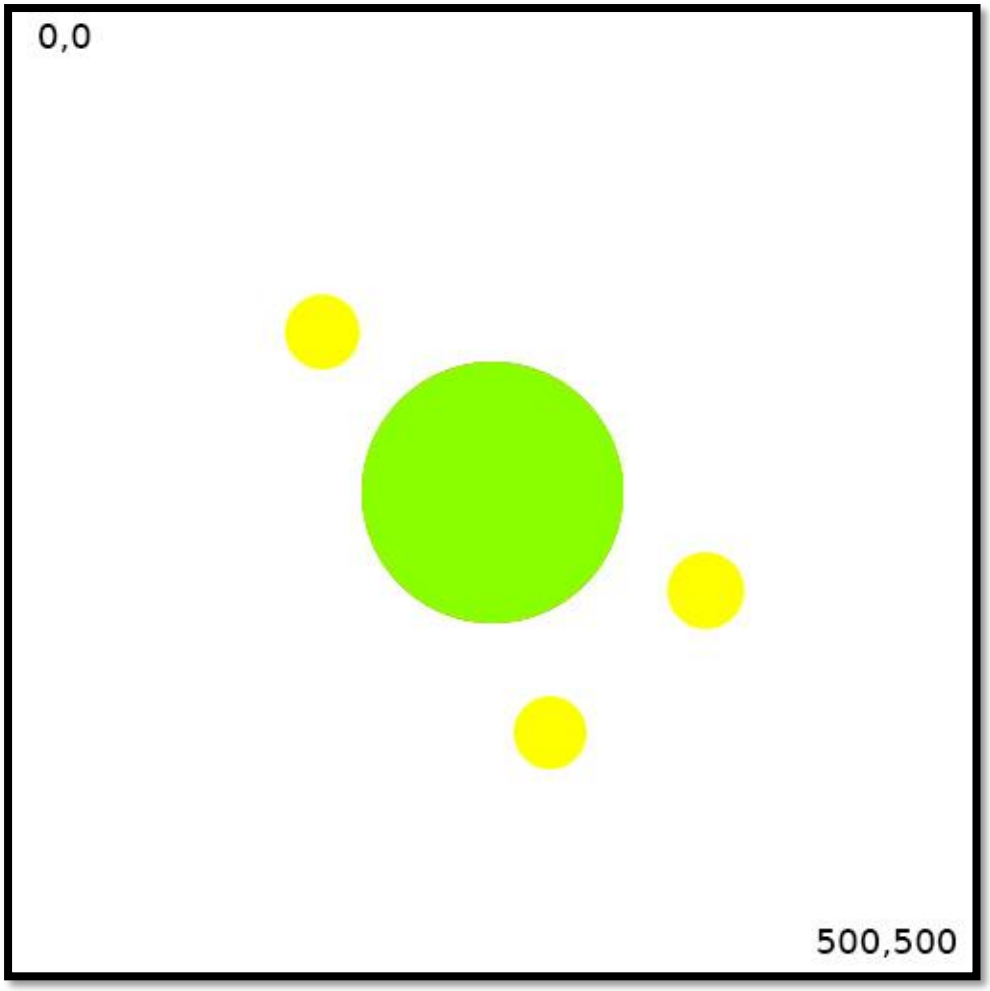
Kun reunat on löydetty, merkitään kaikki reunojen väliset pikselit valituiksi ja lisätään ylempi ja alempi pikseli jonoon (jono merkattu kuvaan vaalean vihreellä).



Sama toimenpide toistetaan niin kauan kunnes jono on tyhjä



Lopputulokseksi saadaan valittu alue, eikä muita pikseleitä tarvitse enää tarkastaa.



Hyvät ja huonot puolet:

+ Nopein, varsinkin suurilla kuvilla

+ Ei valitse mitään turhaa

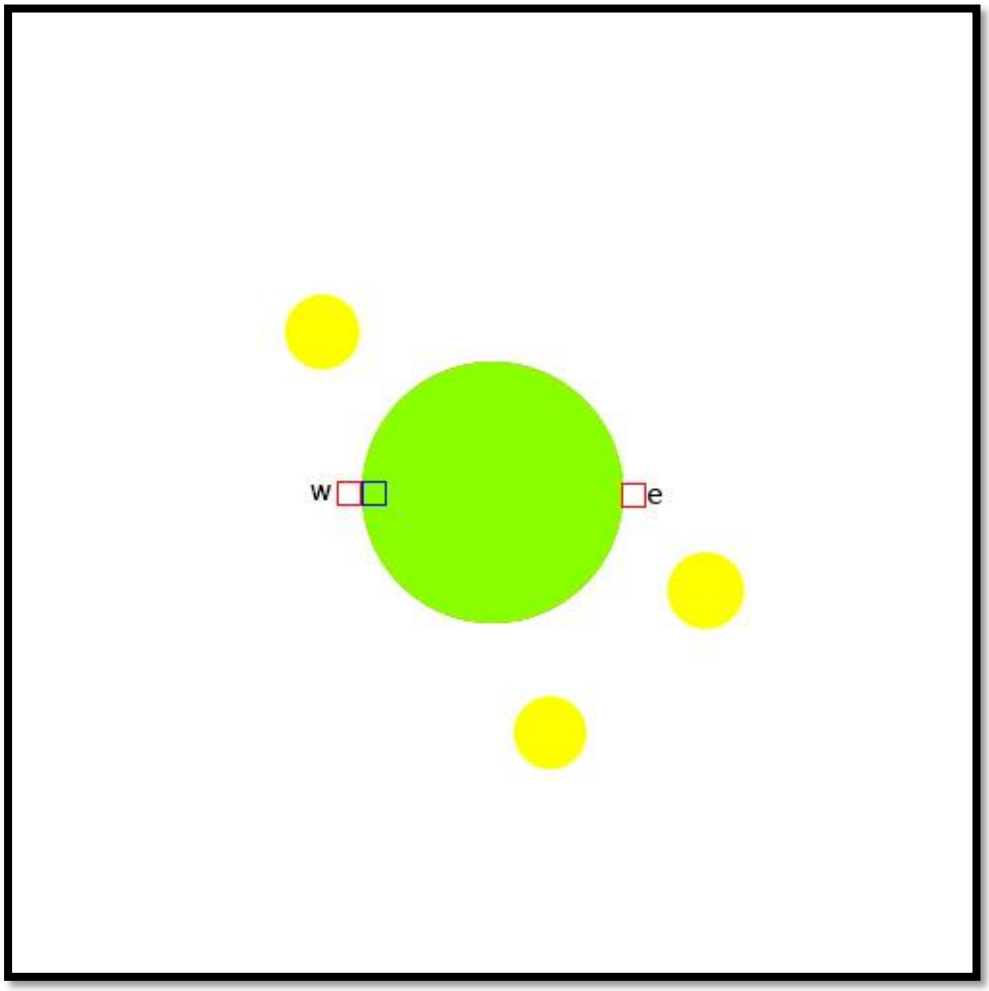
- Epäluotettava, saattaa joskus lähteä "väärille jäljille" jolloin valittu alue ei ole se mitä on toivottu

Replace algoritmit:

Dummy pixel replace

Aika: O(n), Tila: O(n)

Algoritmi käy kaikki valitut pikselit läpi (kuvassa sininen neliö) ja etsii valitun alueen vasemman ja oikean reunan (kuvassa punaiset neliöt w ja e). Sinisen neliön sisältö korvataan jommallakummalla reunalla, riippuen siitä kumpi eroaa valitusta alueesta vähemmän.



Hyvät ja huonot puolet:

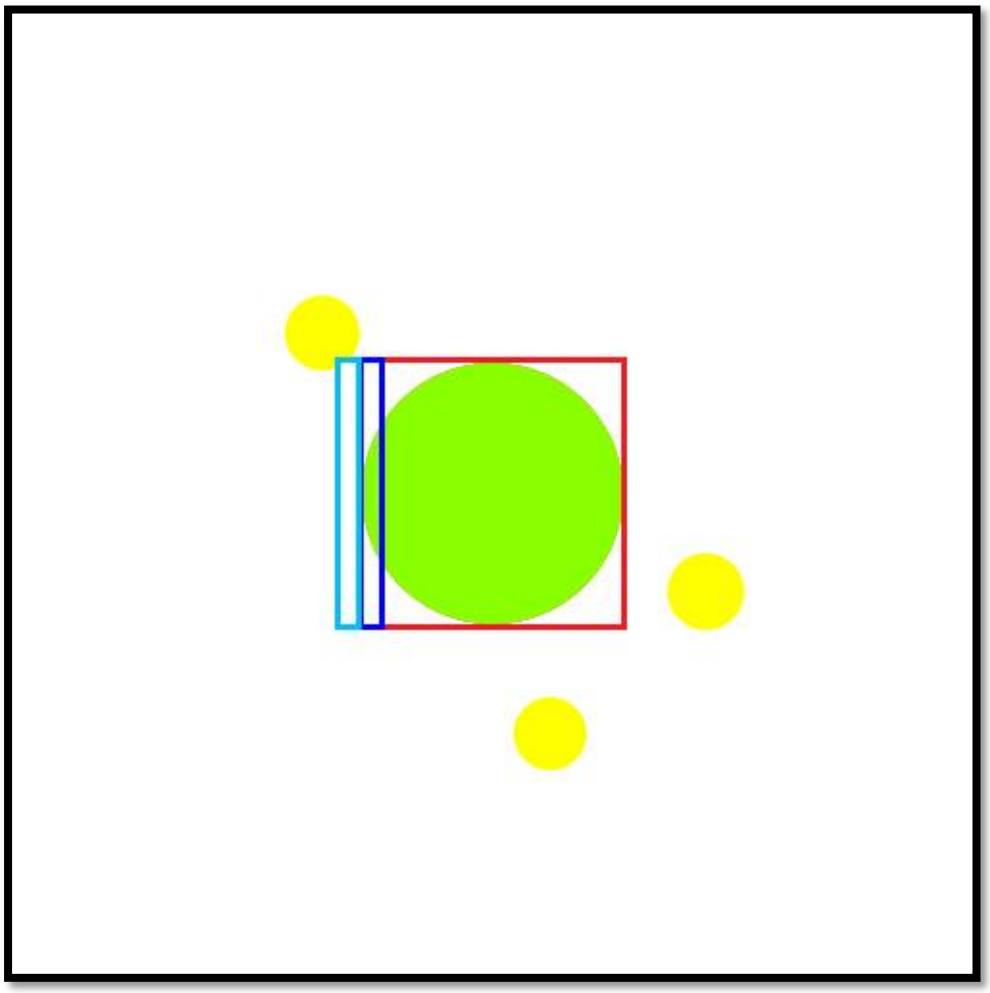
+ Suhteellisen nopea, yksi läpi käynti riittää eikä tarvitse muistia muuhun kuin vasemman ja oikean pikselin muistamiseen

- Laatu ei ole kovin hyvä, varsinkin jos taustan väritys on epätasainen

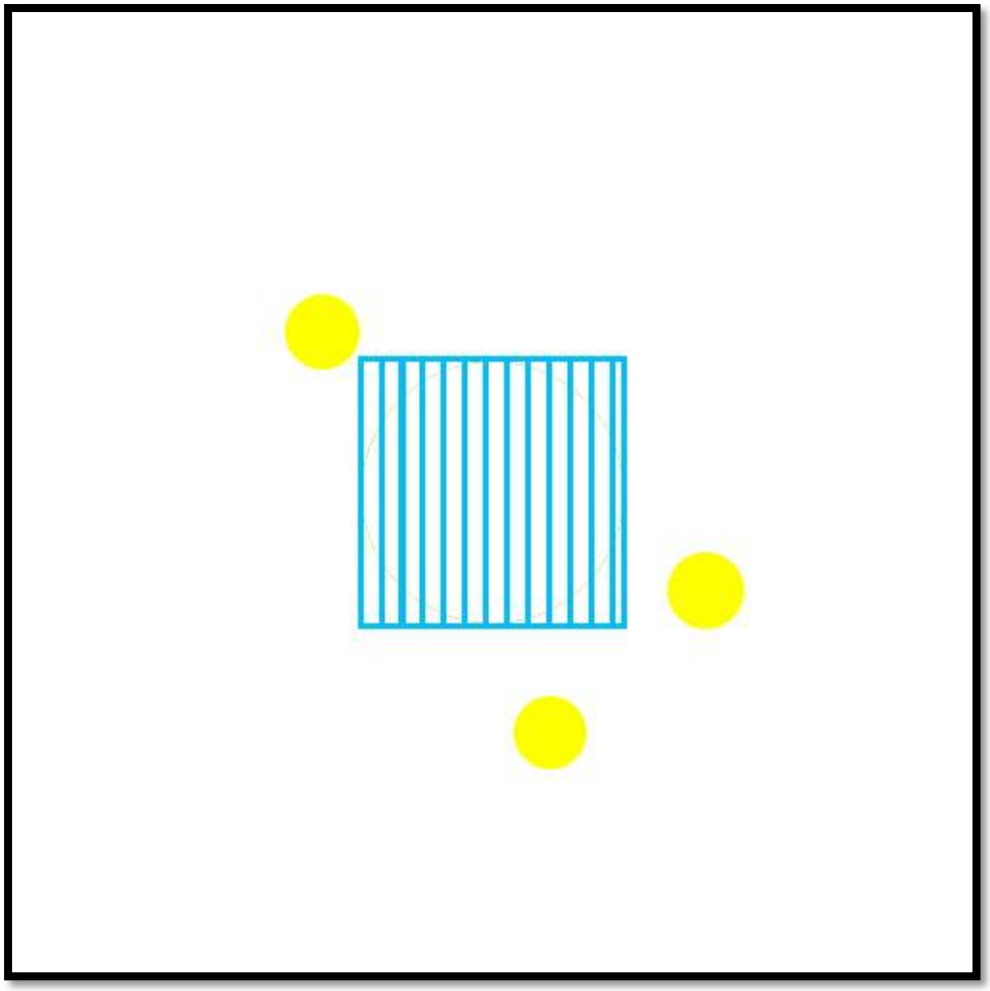
Horizontal repeat

Aika: O(n), Tila: O(n)

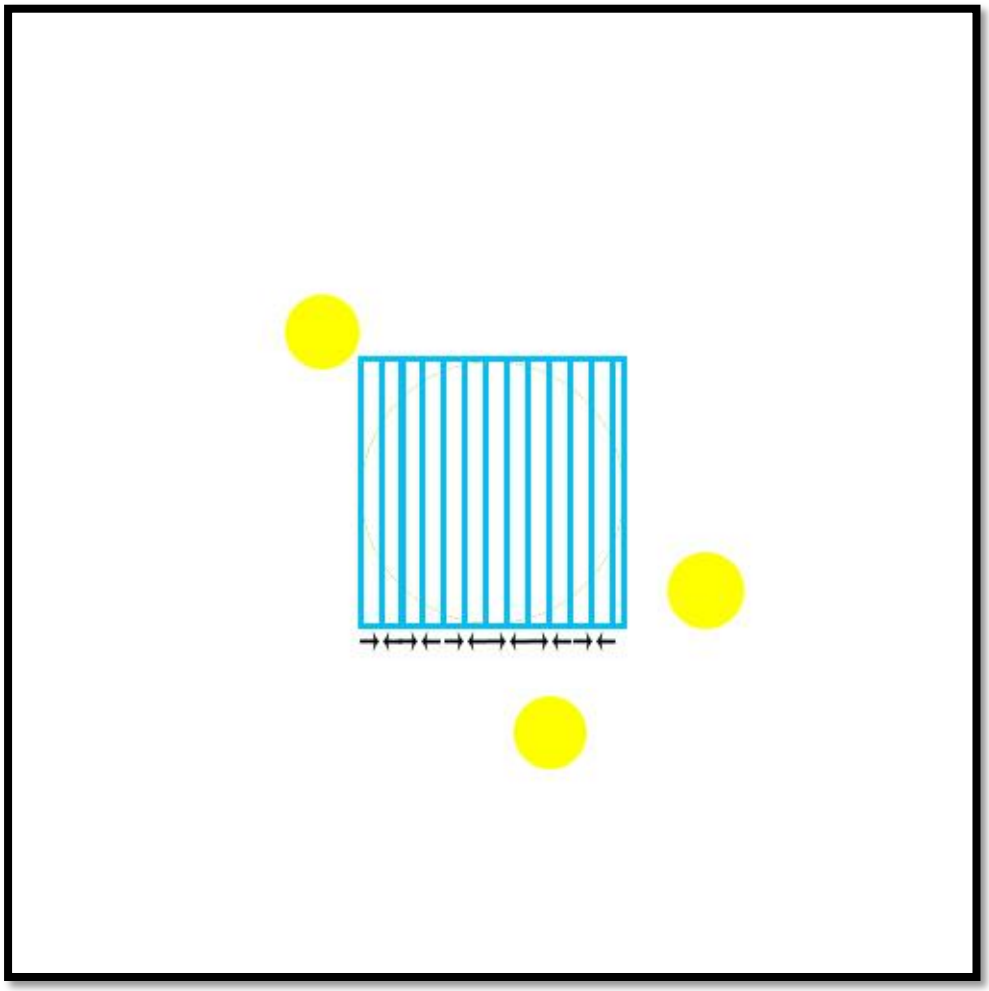
Algoritmi käy kaikki valitut pikselit lävitse. Valitusta alueesta tehdään neliö jonka vasemmasta laidasta alkaen pikseleitä ryhdytään korvaamaan neliön vasemmalla puolella (valitun alueen ulkopuolella) olevilla pikseleillä.



Toimenpide toistetaan kunne koko alue on korvattu.



Algoritmi osaa suorituksen aikana myös kääntää joka toisen valitun alueen peilikuvaksi, jolloin lopputuloksesta tulee saumattomampi.



Hyvät ja huonot puolet:

+ Suhteellisen nopea ja kevyt suorittaa

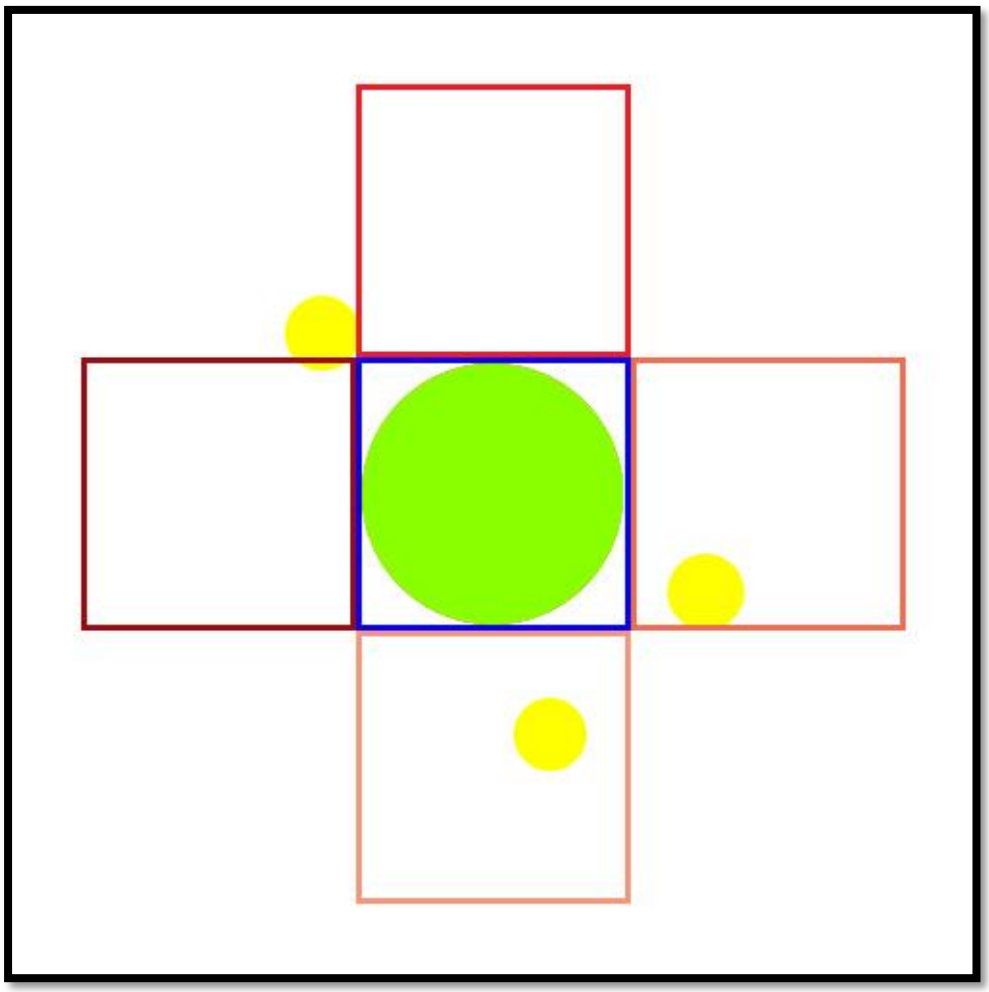
- Ei toimi, jos valittu objekti on kuvan vasemmassa reunassa

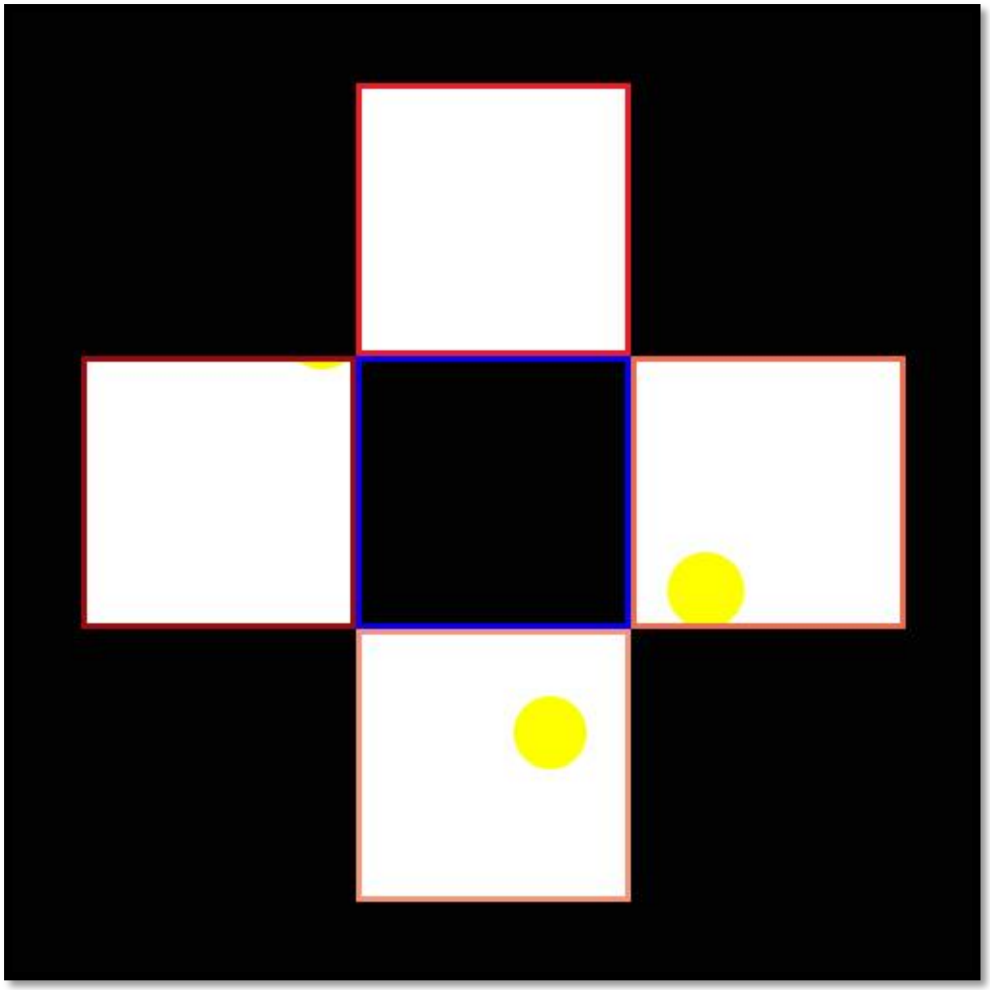
- Lopputuloksessa on parantamisen varaa

Mixed Repeat

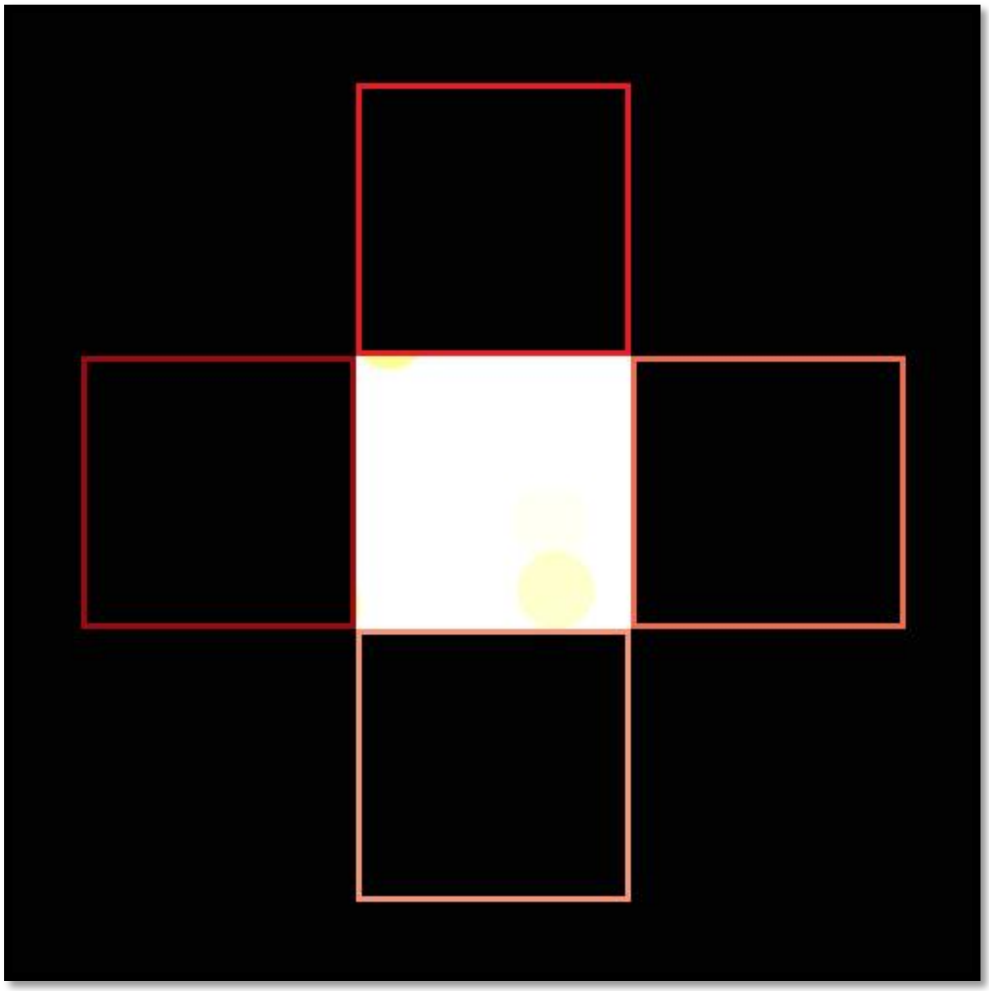
Aika: O(n), Tila: O(n), tosin tilaa tarvitaan <= 4 * valittujen pikseleiden määrä

Tässä algoritmissa valittu alue muutetaan neliöksi, jonka jälkeen samainen neliö valitaan jokaisesta ilmansuunnasta.

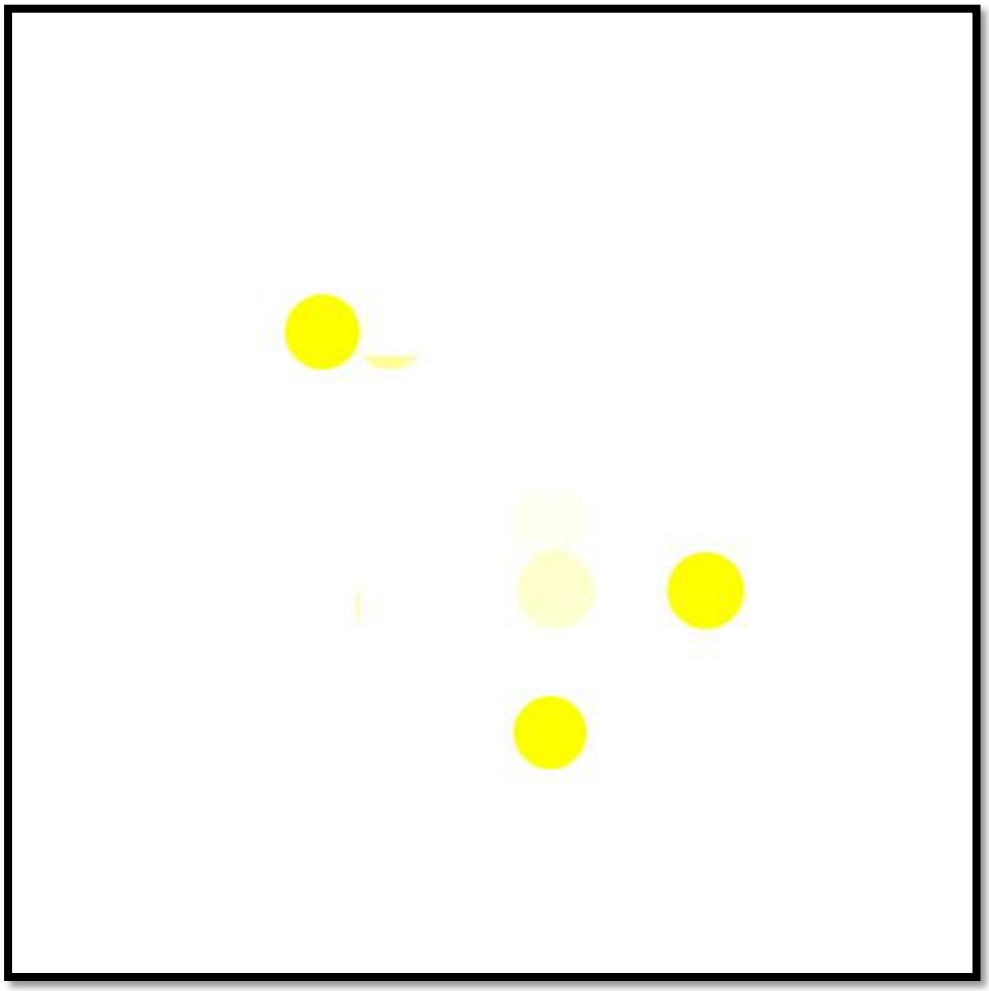




Jokaisen ilmansuunnan neliö peilataan niin että ne jatkuvat omaan suuntaansa mahdollisimman sujuvasti ja keskimmäisen alueen pikseleiden väriksi valitaan neliöiden pikseleiden keskiarvo.



Lopputulokseksi saadaan täytetty alue, joka näyttää ihmissilmään suhteellisen hyvältä, joskin reunat (esim. Kuvassa näkyvä puolikas ympyrä) paistavat melko ikävästi.



Hyvät ja huonot puolet:

+ Ihmissilmälle yleensä paras lopputulos

+ Vain hieman hitaampi kuin muut algoritmit

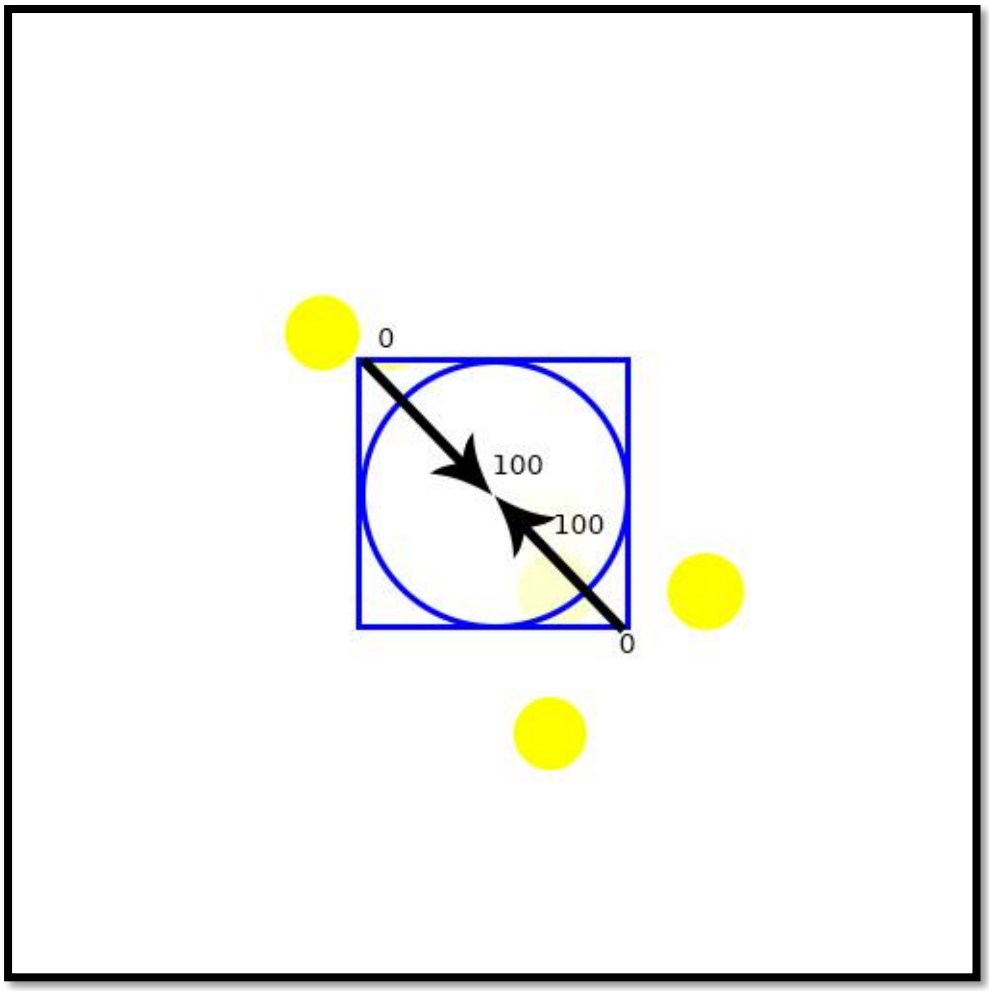
+ Toimii myös vaikkei aluetta saisi valittua kuin yhdestä ilmansuunnasta (tosin aika paljon huonommin)

- Vie hieman enemmän muistia (4 x valittu alue)

- Reunat

Extra:

Lopputuloksen parantamiseksi lisäsin kaavan, joka laskee korvaavalle pikselille alpha arvon pohjautuen sen etäisyyteen valitun alueen keskustasta. Tämän avulla reunat saadaan häivytettyä ja lopputuloksesta tulee entistä parempi!

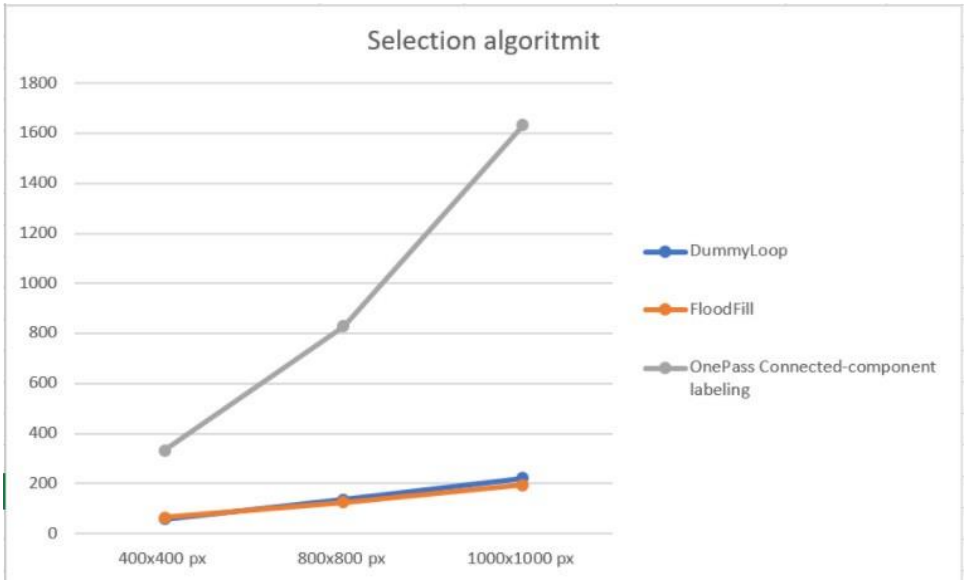


Suorituskykytestit:

Testeissä tehtiin 10 suorituskertaa ja laskettiin niiden keskiarvo.

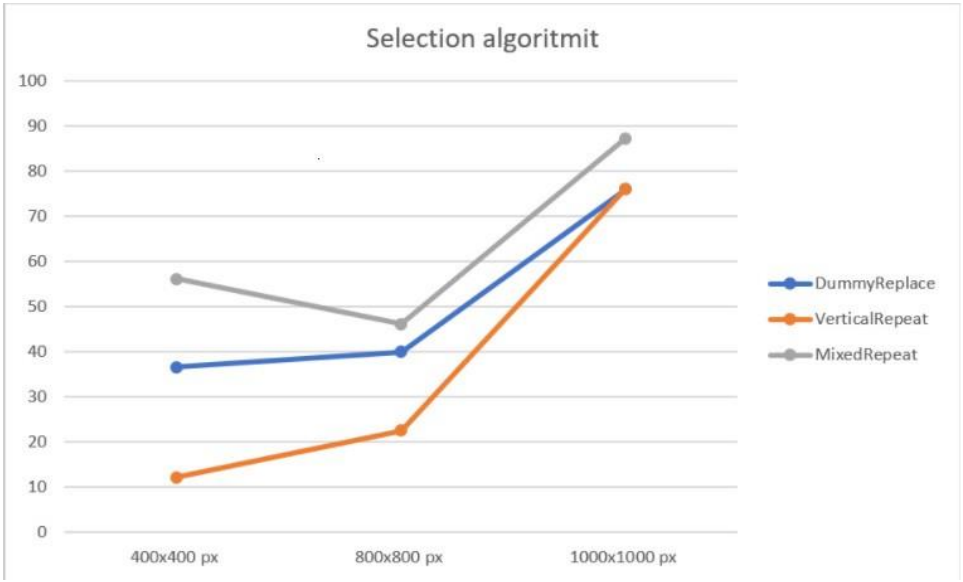
Selection algoritmit:

Algoritmi	400x400 px	800x800 px	1000x1000 px
DummyLoop	**58.6 ms**	136.2 ms	222 ms
FloodFill	66 ms	**125.6 ms**	**195.4 ms**
OnePass Connected-component labeling	333.5 ms	829.6 ms	1631.7 ms



Replace algoritmit: (käytetty FloodFillia valitsemiseen)

Algoritmi	400x400 px	800x800 px	1000x1000 px
DummyReplace	36.6 ms	40 ms	76.3 ms
VerticalRepeat	**12.2 ms**	**22.6 ms**	**76.2 ms**
MixedRepeat	56.1 ms	46.2 ms	87.3 ms



Puutteet ja parannusehdotukset

Todellinen, ns. "pixel perfect" Image inpainting (https://en.wikipedia.org/wiki/Inpainting) on varsin monimutkainen prosessi, joka usein vaatii tuhansia, ellei jopa miljoonia kuvan läpikäyntejä. Tässä työssä keskityttiin nopeuteen ennemmin kuin täydellisen lopputuloksen saavuttamiseen.

Tästä syystä algoritmi ei aina välttämättä toimi juuri toivotulla tavalla, mutta useimmiten haluttu lopputulos voidaan saavuttaa < 10 klikkauksella.

Suurin yksittäinen syy algoritmin epäonnistumiselle on ns. Machine Visionin puuttuminen, eli algoritmin kyky erotella pikseleitä toisistaan. Useissa käytettävistä demokuvista poistettavat alueet koostuvat useista eri värisävyistä, joiden sisäisetkin sävyerot voivat olla merkittävän suuria (esim. valotuksesta johtuen). Tästä syystä algoritmi saattaa usein ajautua tilanteeseen jossa alueen valinta epäonnistuu, jonka seurauksena lopputulos ei ole halutunlainen.

Lisäksi algoritmi tarvitsisi jonkinlaisen ohjausalgoritmin joka määrittelisi algoritmin käyttämät muuttujat kuvan värisävyjen perusteella. Nyt algoritmi toimii muuttujia säätämällä lähes jokaisen kuvan kanssa, mutta on erittäin hankalaa löytää sellaisia arvoja, jotka sopisivat kaikkiin kuviin.

Lähteet

- https://en.wikipedia.org/wiki/Flood_fill
- https://en.wikipedia.org/wiki/Connected-component_labeling
- https://en.wikipedia.org/wiki/Blob_detection
- https://en.wikipedia.org/wiki/Inpainting
- http://www.dtic.upf.edu/~mbertalmio/bertalmi.pdf